

Fatec Praia Grande

Paulo R. T. Cândido

NodeJS - parte 4

Controle de acesso

1) Instalar pacote pra criptografia

```
yarn add bcryptjs
```

```
yarn add @types/bcryptjs -D
```

2) Instalar JWT (Json Web Token), gereador de tokens de segurança

```
yarn add jsonwebtoken
```

```
yarn add @types/jsonwebtoken -D
```

3) Instalar pacote para acessar variáveis em .env

```
yarn add dotenv
```

4) Criar model Usuario

```
model Usuario{
  id Int @id @default(autoincrement())
  email String @db.VarChar(100) @unique
  senha String @db.VarChar(100)

  @@map("usuarios")
}
```

```
yarn prisma migrate dev --name terceira
```

5) Criar rotas (routes.ts) para cadastro de usuários e autenticação

```
// Cadastra usuário
Roteador.post('/usuarios', new UsuarioController().store);

// Autenticação - retorna um token de segurança se usuário for autenticado
Roteador.get('/usuarios/autenticacao', new UsuarioController().autenticacao)
```

6) Gerar chave de segurança para a aplicação e inseri-la como variável de ambiente (arquivo .env)
<https://www.md5hashgenerator.com/>

Relat

• Sha

MD5 Hash Generator

Use this generator to create an MD5 hash of a string:

chaveprojeto123

Generate →

Your String	chaveprojeto123	
MD5 Hash	e340b8b055977280c9f24076ec61c568	Copy
SHA1 Hash	533ade27d30a269179fd1511c23d4d74c0f94a3f	Copy

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like node_modules, prisma, migrations, schema, src, controllers, middlewares, routes, server, and .env. The .env file is selected and highlighted. The code editor shows the content of the .env file, which includes environment variables for Prisma, database URL, and a security key. The security key is highlighted with a red circle and matches the MD5 hash generated in the previous step.

```
1 # Environment variables declared in this file are automatically
2 # See the documentation for more detail: https://pris.ly/d/pri
3
4 # Prisma supports the native connection string format for Post
5 # See the documentation for all the connection string options:
6
7 DATABASE_URL="mysql://root:@localhost:3306/meubd?schema=public"
8
9 # Criar chave de segurança
10 CHAVESEGURANCA=e340b8b055977280c9f24076ec61c568
```

7) Criar UsuarioController com os método store e autenticação

```
import {Request, Response} from 'express';
import {prisma, PrismaClient} from '@prisma/client';
import {hash, compare} from 'bcryptjs'; // pacote de criptografia
import { Secret, sign } from 'jsonwebtoken'; // sign é usado para gerar o token

class UsuarioController {
  async store(req:Request,res:Response){
    const prisma = new PrismaClient();
    const {email, senha} = req.body;
    if (email==null || senha==null) {
      return res.status(400).json({status: 'email e senha devem ser fornecidos.'});
    }
    try {
```

```

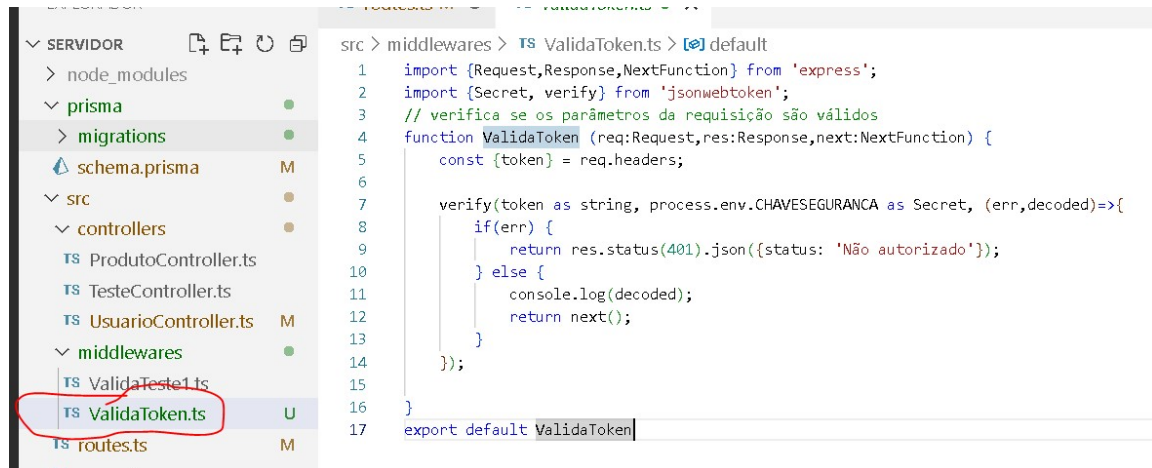
    const novoUsuario = await prisma.usuario.create(
      {
        data: {
          email: email,
          senha: await hash(senha,8), // criptografa a senha, 8 é o salto
        },
        select: {
          email: true
        }
      }
    );
    res.status(200).json(novoUsuario);
  }
  catch (erro){
    return res.status(400).json({status: 'email deve ser único'});
  }
}

async autenticacao(req:Request,res:Response){
  const prisma = new PrismaClient();
  const {email, senha} = req.body;
  const consulta = await prisma.usuario.findFirst(
    {
      where: {
        email: email
      }
    }
  );
  if (consulta==null){
    return res.status(401).json({status: 'não autorizado'});
  } else {
    console.log(consulta.senha);
    console.log((await hash(senha,8)).length);
    if (await compare(senha,consulta.senha)) { // senha bate
      // gerar token
      const token = sign(
        {
          email: consulta.email
        },
        process.env.CHAVESEGURANCA as Secret,
        {
          subject: consulta.id.toString(),
          expiresIn: '1d'
        }
      );
      return res.status(200).json({token:token});
    } else {
      return res.status(401).json({status: 'não autorizado'});
    }
  }
}
}

export default UsuarioController

```

8) Criar middleware para validação do token gerado na autenticação, usá-lo para restringir acesso ao cadastro de produtos



```
import {Request,Response,NextFunction} from 'express';
import {Secret, verify} from 'jsonwebtoken';
// verifica se os parâmetros da requisição são válidos
function ValidaToken (req:Request,res:Response,next:NextFunction) {
    const {token} = req.headers;

    verify(token as string, process.env.CHAVESEGURANCA as Secret, (err,decoded)=>{
        if(err) {
            return res.status(401).json({status: 'Não autorizado'});
        } else {
            console.log(decoded);
            return next();
        }
    });
}
export default ValidaToken
```

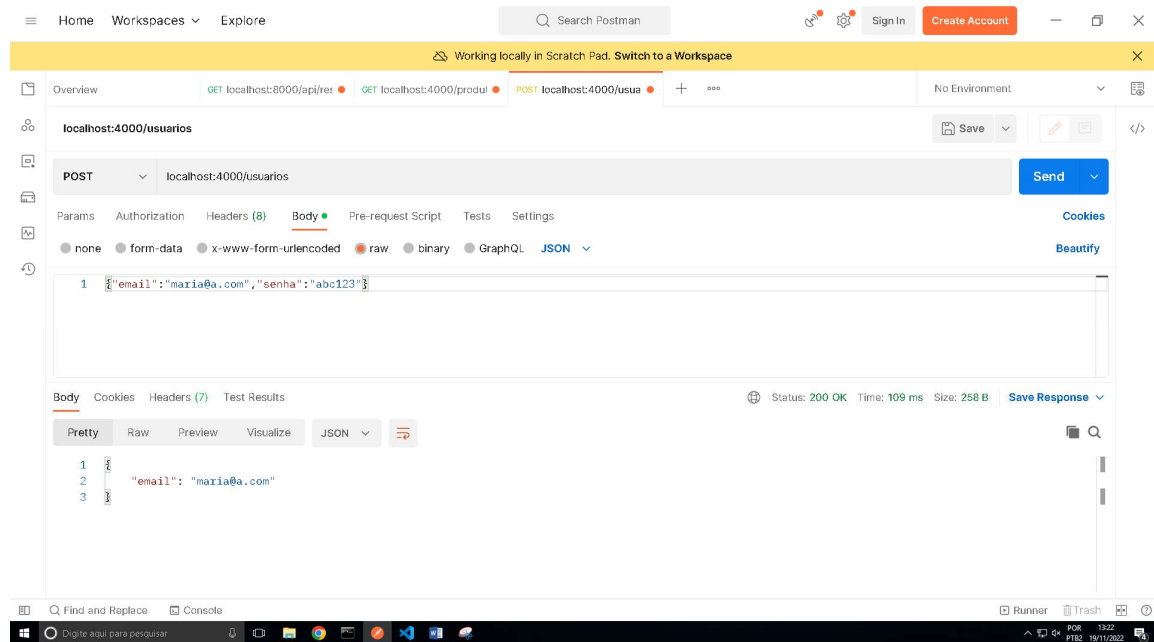
9) Incluir o middleware de validação do token nas rotas de produtos, restringindo acesso a usuários autenticados.

```
import ValidaToken from './middlewares/ValidaToken';

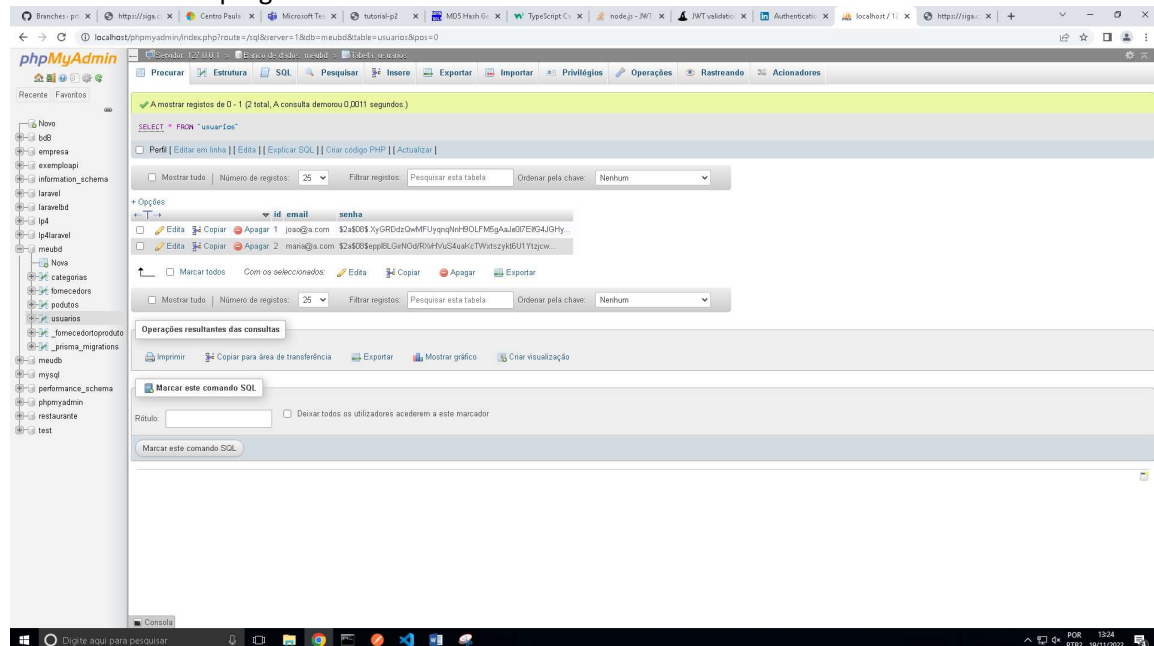
Roteador.get('/produtos', ValidaToken, new ProdutoController().index);
Roteador.get('/produtos/:id', ValidaToken, new ProdutoController().show);
Roteador.post('/produtos', ValidaToken, new ProdutoController().store);
Roteador.put('/produtos/:id', ValidaToken, new ProdutoController().update);
Roteador.delete('/produtos/:id', ValidaToken, new ProdutoController().delete);
Roteador.put('/produtos/fornecedores/:id', ValidaToken, new
ProdutoController().associarFornecedores);
```

10) Realizar testes

Cadastrar novo usuário



Verificar senha criptografada



Tentar autenticar com senha inválida

Home Workspaces Explore Search Postman Sign In Create Account

Working locally in Scratch Pad. Switch to a Workspace

Overview GET localhost:8000/api/res GET localhost:4000/produti GET localhost:4000/usuario + ... No Environment

localhost:4000/produtos Save

GET localhost:4000/produtos Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Headers 8 hidden

KEY	VALUE	DESCRIPTION	Bulk Edit	Presets
<input checked="" type="checkbox"/> token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFnepC6Imthcm...			
Key	Value	Description		

Body Cookies Headers (7) Test Results Status: 200 OK Time: 50 ms Size: 299 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "nome": "Calha",
3   "preco": "36.7",
4   "categoria": {
5     "nome": "Básico"
6   }
7 }
8 }
```

Find and Replace Console Runner Trash

Modificar o token (1ª letra) para forçar falha de acesso

Home Workspaces Explore Search Postman Sign In Create Account

Working locally in Scratch Pad. Switch to a Workspace

Overview GET localhost:8000/api/res GET localhost:4000/produti GET localhost:4000/usuario + ... No Environment

localhost:4000/produtos Save

GET localhost:4000/produtos Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Headers 8 hidden

KEY	VALUE	DESCRIPTION	Bulk Edit	Presets
<input checked="" type="checkbox"/> token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFnepC6Imthcm...			
Key	Value	Description		

Body Cookies Headers (7) Test Results Status: 401 Unauthorized Time: 12 ms Size: 273 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": "Não autorizado"
3 }
```

Find and Replace Console Runner Trash

Exercício: testar outras operações com produtos.

